Advanced Metering Infrastructure (Smart Meter) Big Data Analysis and Energy Demand Forecasting

Isaac Salvador
UIN: 669845132

December 5th, 2023
IDS 561 Big Data Analytics
Professor Yuheng Hu

Contents

Chapter 1 Introduction

1.1 Project Context

On December 19th, 2007, the Energy Independence and Security Act (EISA) of 2007 was passed by the U.S. Congress, aimed at addressing U.S. energy security, renewable energy development, and improving vehicle fuel economy. A key aspect of the Act was legislation that promoted the transformation of the U.S.'s aging energy infrastructure into a "smart grid". The U.S. Department of Energy notes that smart grids incorporate "cutting-edge technologies, equipment, and controls that communicate and work together to deliver electricity more reliably and efficiently".

The proliferation of advanced metering infrastructure (AMI) and other grid monitoring and communication technologies in the mid 2000's has led to the accumulation of vast amounts of big data related to smart grids and energy consumption. Colloquially known as "smart meters" these devices serve an integral part in smart grid operations and produce a plethora of energy consumption data for analysis. This project leverages the advances made in deep learning and big data analytics to analyze data obtained from smart meters owned and operated by Commonwealth Edison (ComEd), the largest electric utility company in Illinois, and the primary power provider for the City of Chicago.

The objective of this research is to develop a big data pipeline to assess smart meter energy consumption data sourced from ComEd resources. The pipeline will begin with collecting disparate smart meter data collected in January, 2021, for ComEd customers located in Chicago, Illinois. The project will utilize **MongoDB** for data storage and handling, and use deep learning techniques to forecast energy demand from the extracted data.

1.2 Literature Review

Big data analytics has already found application in numerous aspects of the energy industry. Daki et al.[1] outlines the requirements and implementation of managing big data related to smart grid operations. This data can be obtained from sensors, substations, AMI smart meters, control centers, and customer accounts and devices. Their study recommends the use of Hadoop for batch processing and Spark for real-time processing.

In terms of load forecasting big data also plays an important role. Ordiano et al.[2] benchmarked the performance of Spark against single computer approaches past a threshold training set comprised of $1.57 \cdot 10^7$ data instances. A study by Perçuku et al. implemented a NoSQL data store and time series methods to predict short term load demand. Matthews and Leger[3] showed the successful application of the MapReduce framework to analyze time series synchrophasor data for real-time smart grid anomaly detection.

Time series data generated from energy consumption is also a prime candidate for machine learning and deep learning techniques. A study by Farsi et al.[4] developed a hybrid long short-term memory (LSTM) and convolutional neural network (CNN) model dubbed PLCNet for short-term load forecasting that achieved 98.23% accuracy on hourly energy demand test data.

Chapter 2 Data

2.1 Energy Demand Time-Series Data

The primary data source for this research originates from ComEd's *Anonymous Data Service*. This service provides up to 24 months' access to time-series interval energy usage data on a meter-per-meter basis. Data for the service is available for smart meters installed in ComEd's operating territory and instances of the data have been grouped by customer ZIP code.

The data is provided in the form of data sets aggregated at the ZIP code label per month of recorded data. ComEd currently offers two products for this service: a five-digit ZIP code offering (ZIP), and a nine-digit (ZIP+4) offering. This research uses the former product and encompasses the ZIP codes belonging to Chicago, Illinois. For this research, time-series data from the month of January 2021 was used for analysis. The total size of this data was approximately 70 GB and individual `.csv` files were on average 200 MB each.

2.2 Anonymization Protocols

The data provided by ComEd is anonymized to not include any customer specific personal identifiers and ensures confidentiality of customer-specific information. In particular, the data follows the "*15/15*" anonymization protocols established by the Illinois Commerce Commission (ICC).

The protocol states that each data set must contain at least 15 customers, and not one of the 15 can exhibit 15% or more of the usage within that data set. This measure in addition to removing personally identifiable information are a part of the requisite steps necessary to maintain anonymity and privacy within data sets and for the entire product offering.

2.3 Delivery Format and Data Structure

The data is delivered to requestors in the form of compressed `.csv` files with the typical file-name `ANONYMOUS_DATA_YYYYDD_ZIPCODE.csv`. One file is equivalent to one ComEd data set, and contains interval energy use for one zip code for one recorded month. Data is formatted tabularly, with each row in a data set corresponding to one customer located in the file name's ZIP code for one day in the indicated month. Table 2.1 shows sample instances of one customer's smart meter data for 15 days from the file `ANONYMOUS_DATA_202101_60607.csv` with the following field names and descriptions:

(1) **ZIP** (`"ZIP_CODE"`): The ZIP code of the customer.
(2) **CLASS** (`"DELIVERY_SERVICE_CLASS"`): Customer service level (single residential, multi-family residential, commercial, etc.).
(3) **ACCOUNT** (`"ACCOUNT_IDENTIFIER"`): Randomized identification number for an individual customer. Due to ICC anonymization protocols, identification numbers are identical within a data set but are inconsistent among all data sets.
(4) **DATE** (`"INTERVAL_READING_DATE"`): Date of smart meter recording.
(5) **TOTAL** (`"TOTAL_REGISTERED_ENERGY"`): Total recorded energy consumed by a customer in Kilowatt-hour (kWh).

(6) **HRXXXX** (`"INTERVAL_HRXXXX_ENERGY_QTY"`): 30-minute interval smart meter recording in kWh.

TABLE 2.1. Sample Time-Series Data (Select Fields)

ZIP	CLASS	ACCOUNT	DATE	TOTAL	HR0030	···	HR2400
60607	C23	10019179914695	1/1/21	10.0435	0.1975	···	0.1700
60607	C23	10019179914695	1/2/21	11.8411	0.1875	···	0.1500
60607	C23	10019179914695	1/3/21	11.0432	0.1775	···	0.1225
60607	C23	10019179914695	1/4/21	9.6046	0.2325	···	0.1400
60607	C23	10019179914695	1/5/21	15.0873	0.1800	···	0.2575
60607	C23	10019179914695	1/6/21	18.1357	0.27	···	0.195
60607	C23	10019179914695	1/7/21	14.8153	0.1975	···	0.275
60607	C23	10019179914695	1/8/21	16.6922	0.325	···	0.345
60607	C23	10019179914695	1/9/21	18.5246	0.3425	···	0.3475
60607	C23	10019179914695	1/10/21	15.9393	0.2625	···	0.1925
60607	C23	10019179914695	1/11/21	15.8508	0.415	···	0.425
60607	C23	10019179914695	1/12/21	13.3745	0.2	···	0.15
60607	C23	10019179914695	1/13/21	14.559	0.3575	···	0.1925
60607	C23	10019179914695	1/14/21	15.9374	0.1875	···	0.245
60607	C23	10019179914695	1/15/21	17.5988	0.28	···	0.215

2.4 Geospatial Data

In addition to time-series data, auxiliary data in the form of Geographic Information Systems (GIS) shapefiles (`.shp`) were used for spatial analysis and plotting. Shapefiles store vectorized geographic feature data with accompanying tabular attributes for each feature in the `.shp` file. While industry applications of this type of data typically use proprietary software such as ESRI *ArcGIS*, spatial operations for this research was performed using the python library `geopandas`, a `pandas`-like library that extends typical data analysis tools to spatial data types.

The source of this geospatial data comes from the City of Chicago's Open Data Portal, a repository for open-source data maintained by the city. The `.shp` file contains the extent of the City of Chicago, subdivided into it's requisite ZIP code boundaries. Figure 2.1 shows the project area as described above.

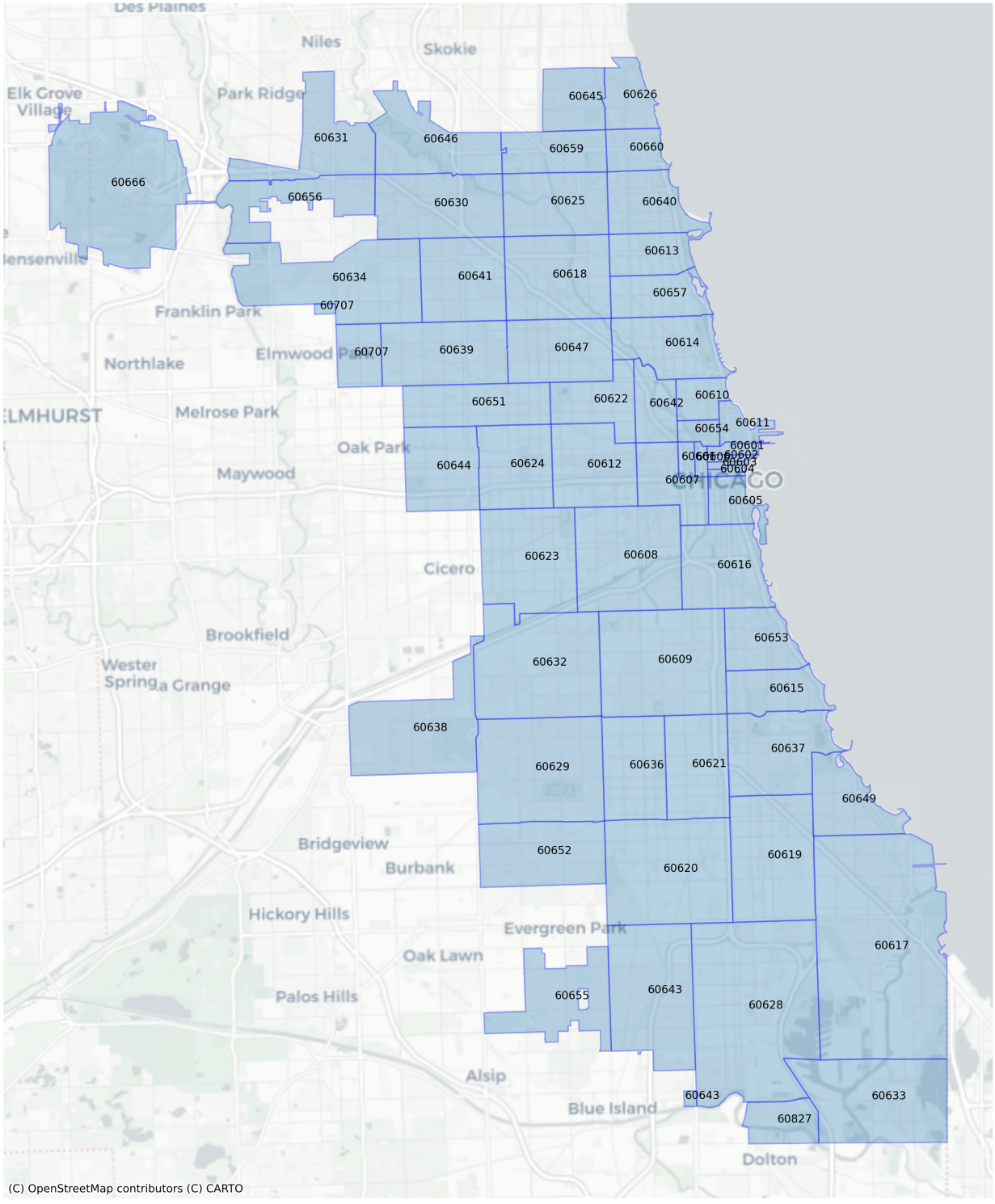

FIGURE 2.1. Research Project Limits and ZIP Codes

Chapter 3 Big Data

3.1 MongoDB

Due to the large volume of data used and the nascent nature of this research, the distributed database **MongoDB** was used for data management and development. In addition to database features, the **MongoDB** ecosystem includes online SaaS tools for monitoring and a flexible python API `pymongo` for pipeline integration. These features allowed for the rapid development of a database system for time-series interval data.

The primary advantage of **MongoDB** is its NoSQL framework known for flexibility and scalability, and the use of a "document" based data model. A document is the fundamental storage unit of **MongoDB**, and is composed of key-value pairs stored in a JSON-like structure called a BSON (Binary-JSON). Unlike traditional relational database management (RDBM) systems, documents are allowed to have flexible schema, accommodating for a wide range of data structures within the database. Figure 3.1 shows an equivalent BSON document of the first instance of Table 2.1

```
{
    "_id": ObjectId("6562c7258101a3cc3d8bce30"),
    "ZIP_CODE": 60607
    "DELIVERY_CLASS": "C28",
    "ACCOUNT_IDENTIFIER": 10019179914695,
    "INTERVAL_READING_DATE": 2021-01-01T00:00:00.000+00:00,
    "TOTAL_REGISTERED_ENERGY": 10.0435,
    "INTERVAL_HR_0030_ENERGY": 0.1975,
    ...
    ...
    "INTERVAL_HR_2400_ENERGY": 0.17
}
```

FIGURE 3.1. Sample **MongoDB** BSON Document

An API wrapper was used in conjunction with `pymongo` to collect the raw `.csv` files and upload them into a **MongoDB** collection. The wrapper first filtered the input files for the ZIP codes belonging to the City of Chicago, then made the appropriate data type conversions such as string to float and date-time. Upon type conversion, the wrapper utilizes the collection method `insert_many()` to upload the data to **MongoDB**. This process took approximately twenty hours to complete and was performed in two passes.

3.2 Data Aggregation

Once uploaded onto remote cloud services, **MongoDB**'s aggregation pipeline was used to process and transform the time-series interval data. The aggregation pipeline consists of

multiple stages specified by operations or transformations to achieve the desired aggregation results. Unlike the MapReduce framework, the aggregation pipeline is more memory efficient and supports real-time data processing. These gains in efficiency is attributed to **MongoDB**'s dynamic indexing of document instances, enhancing aggregation execution and data retrieval.

In the context of this research, the aggregation pipeline was facilitated using the `pymongo` collection method `aggregate()`, whose argument is a list of dictionaries specifying the operations to be performed on the database collection. Figure 3.2 shows the specific aggregation pipeline operations, namely the `"$match"` operator and the subsequent `"$group"` operator. Using python, this aggregation task took less than 15 minutes to execute on over 40,000,000 documents in the **MongoDB** collection.

```
pipeline = [
    {
        "$match": {
            "INTERVAL_READING_DATE": {
                "$gte": start_day,
                "$lt": end_day + timedelta(days=1)
            },
            "ZIP_CODE": {"$in": zip_codes}
        }
    },
    {
        "$group": {
            "_id": {
                "ZIP_CODE": "$ZIP_CODE",
                "month": {"$month": "$INTERVAL_READING_DATE"},
                "day": {"$dayOfMonth": "$INTERVAL_READING_DATE"}
            },
            "total_energy": {"$sum": "$TOTAL_REGISTERED_ENERGY"},
            "HR0030": {"$sum": "$INTERVAL_HR0030_ENERGY_QTY"},
            ...
            "HR2400": {"$sum": "$INTERVAL_HR2400_ENERGY_QTY"},
        }
    }
]
```

FIGURE 3.2. **MongoDB** Aggregation Pipeline

3.3 Aggregation Results

Aggregation begins by filtering documents with the `"$match"` operator, specifying a single day in January 2021 for a specific ZIP code. Instances of filtered data are then collected using the `"$group"` operator. Energy consumption data is finally summed together for the

appropriate groupings using the `"$sum"` operator. The output of this aggregation pipeline was performed in `pymongo` and was saved locally with a python API wrapper. Table 3.1 shows the results of this operation which was then used for subsequent analysis and energy demand forecasting.

TABLE 3.1. Aggregated Interval Time-Series Data

ZIP_CODE	month	day	total_energy	HR0030	···	HR2400
60645	1	29	423530.1985	8140.3524	···	7612.6729
60622	1	24	893940.9537	17180.7323	···	16274.0598
60605	1	23	1712009.966	36719.3554	···	34777.9125
60624	1	2	415255.3538	8549.1965	···	8062.2883
60622	1	11	966370.0708	16109.21125	···	16974.23706
60649	1	6	514195.4098	10417.80666	···	10400.87354
60629	1	22	890834.9641	16393.74451	···	19115.16626
60610	1	27	1975681.957	34982.67328	···	38270.93901
60611	1	17	3311626.969	61367.83522	···	65876.26467
60654	1	28	481274.8033	9115.835082	···	9236.88474
60614	1	20	1502312.272	26372.34176	···	25939.33884
60621	1	23	335706.2369	6962.419837	···	6826.69534
60649	1	10	534538.6872	10680.36165	···	11090.41307
60639	1	19	977070.8832	16915.00264	···	17650.46533
60605	1	18	1626810.112	31993.2932	···	32718.688
60637	1	28	555022.3594	11090.98732	···	11096.18239
60611	1	6	3420101.034	63554.21114	···	61867.023
60654	1	10	436370.9094	8374.55102	···	8595.25268
60621	1	29	328044.9048	6834.808989	···	6589.01748
60608	1	1	1027927.055	20721.97351	···	20676.92196
60707	1	26	439517.7032	7825.823397	···	8200.919296
60659	1	29	383221.3982	7299.963492	···	7305.805984
60603	1	16	695909.2109	12809.5753	···	12702.7782
60655	1	5	269836.0017	5021.022487	···	5054.695432

Chapter 4 Analysis Methods

4.1 Energy Consumption Patterns

Exploratory data analysis (EDA) was conducted on the reduced output data set obtained in Chapter 3 to assess the performance of the **MongoDB** aggregation pipeline. This EDA consists of visualization of the time series data across temporal granularities, with additional geospatial context.

Plotting daily total energy use from `"total_energy"` recorded between January 1st, to January 31st, 2021, daily energy consumption behavior recorded for the City of Chicago can be displayed at a temporal granularity of once every 24 hours. Figure 4.1 shows this history for the record period, contextualized by consumption per ZIP code.

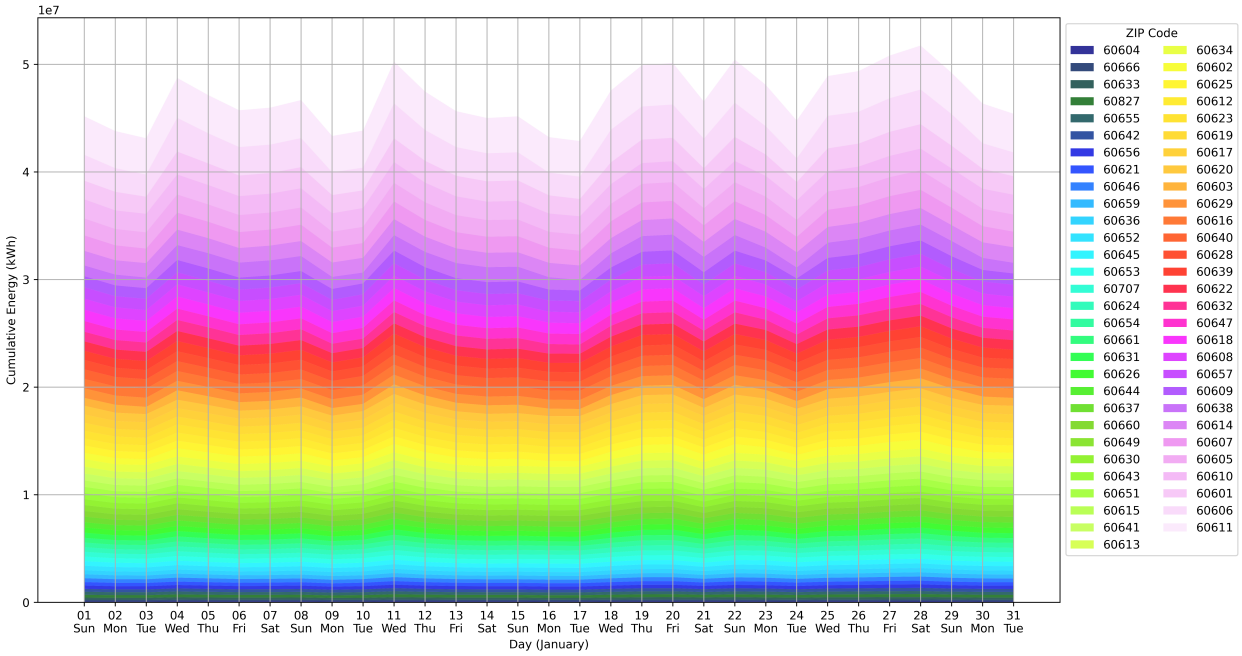

FIGURE 4.1. Stack Plot of Daily Energy Consumption (Geospatial Context)

The first two weeks of January showed a consistent cyclical pattern of daily energy use, characterized by weekly high consumption on Wednesdays, with an additional off-peak on Sundays. The last two weeks however, exhibited irregular consumption history with no discernible pattern, with peak monthly consumption of over 50,000,000 kWh on January 28th. Unsurprisingly, the top five ZIP codes by daily consumption are those located in the Loop and River North neighborhoods of Chicago.

Focusing attention on the 28th, Figure 4.2 shows the monthly peak consumption via choropleth mapping of the research project limits. Indeed, a large proportion of energy consumption occurs in Chicago's central business district along with pronounced energy use in the ZIP code areas surrounding the I-55 and I-90/94 transportation corridors.

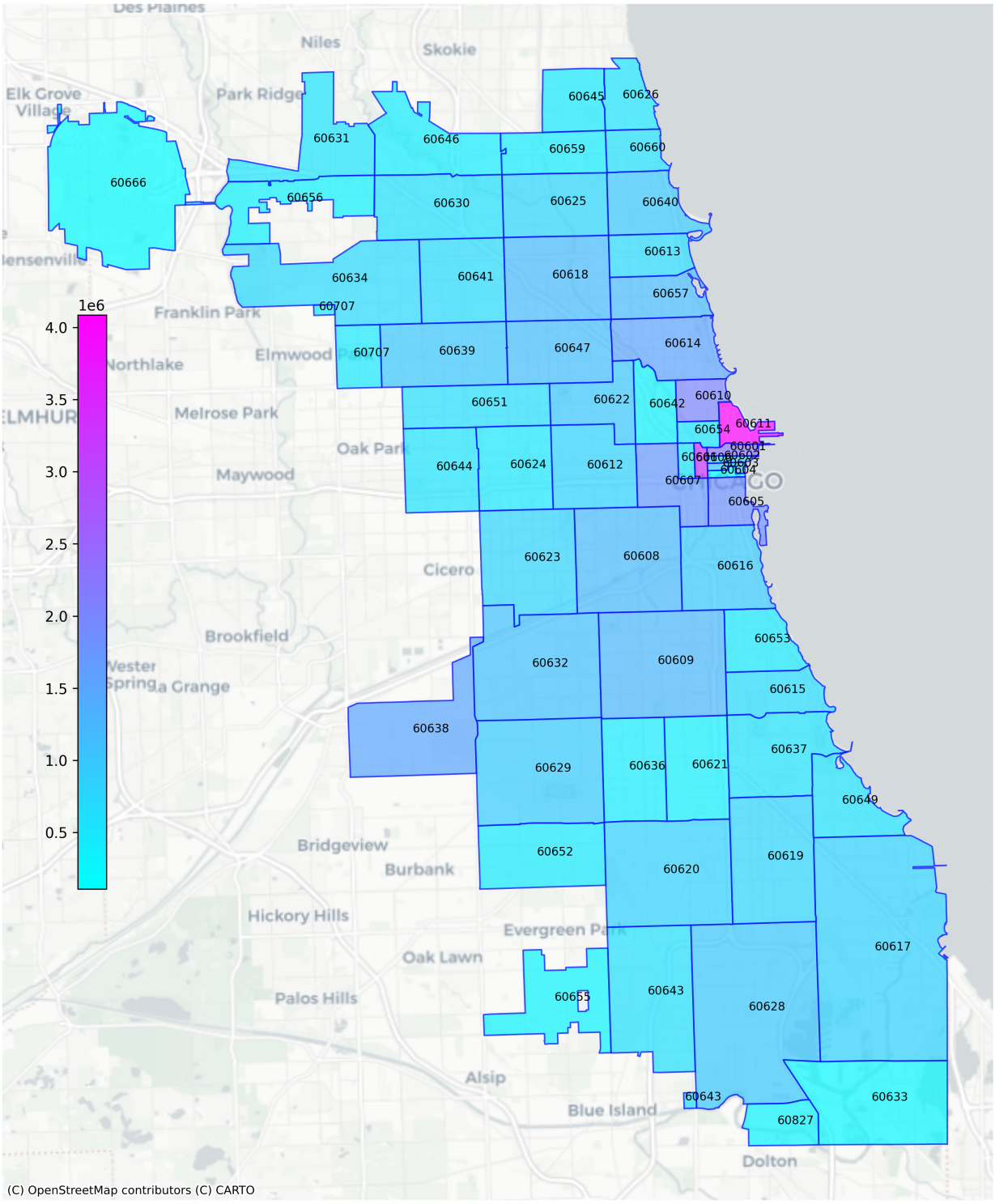

FIGURE 4.2. Choropleth Mapping of Daily Energy Consumption (kWh)

The **MongoDB** aggregation pipeline allows for the investigation of energy consumption at the semi-hourly level. Plotting this consumption history with the same spatial context as Figure 4.1, Figure 4.3 illustrates semi-hourly consumption for an entire 24-hour use-cycle. Temporal granularity at the semi-hourly level provides additional information on consumption behavior, and the figure exhibits the typical energy demand curve characteristics for a 24-hour cycle.

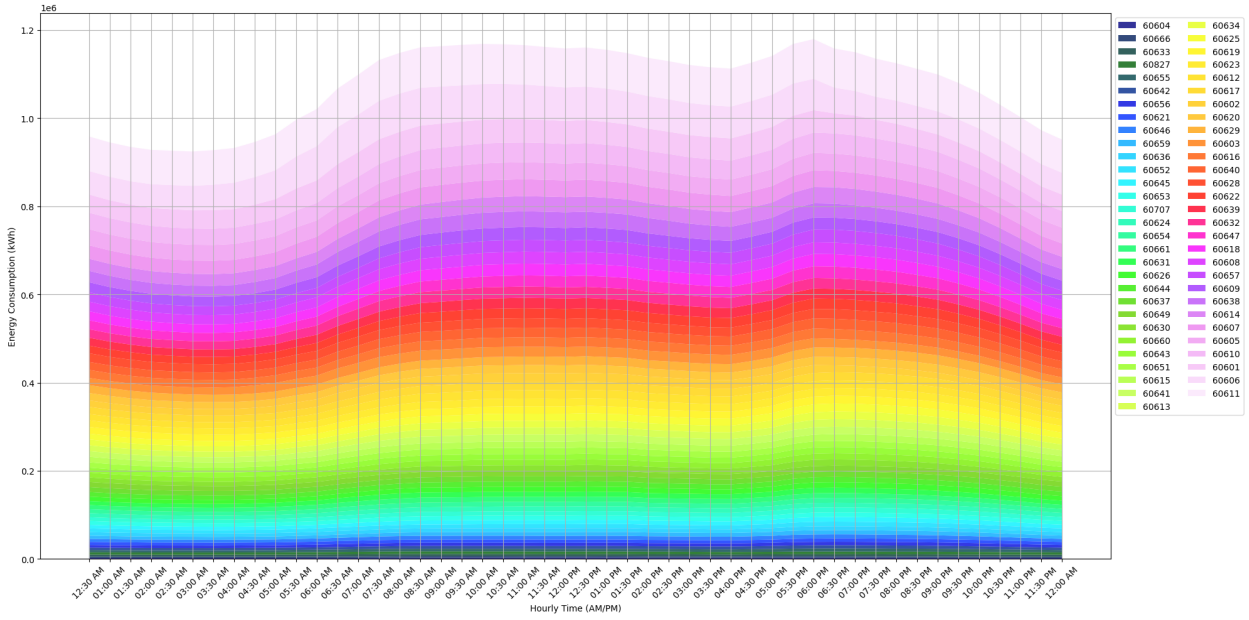

FIGURE 4.3. Stack Plot of Semi-Hourly Energy Consumption

Figure 4.3 illustrates a nightly low consumption at 3:00 AM, followed by peak morning energy consumption at 10:00 AM the following morning. There is an additional, pronounced evening peak that occurs at 6:00 PM followed by the evening "shoulder period". While not the focus of this research, it is pertinent to discuss that the data recorded for this research occurred during a peak COVID-19 lockdown period. Peak morning consumption occurs noticeably later than pre-2020 consumption trends, consistent with prior research[5] for energy consumption in 2021 during the COVID-19 pandemic in North America.

4.2 Weekly Consumption Patterns

The daily behavior exhibited in Figure 4.3 does not solely occur on the 28$^{\text{th}}$. Extending the semi-hourly analysis to the last week of January 2021, daily patterns of energy consumption are similar for every day of the week. From this observation, weekly trends in energy consumption can then be explored. Figure 4.4 shows this semi-hourly consumption between January 25$^{\text{th}}$ to January 31$^{\text{st}}$.

Figure 4.4 shows pattern similarities between daily energy use, with increasing total use up to Thursday, the 28$^{\text{th}}$. For the remainder of the week total energy consumption decreases, with the final day deviating contextually, missing a morning peak. Interestingly, a majority of days exhibited point anomalous energy use during the evening shoulder period. At the

time of writing this report, the cause of these consumption spikes have yet to be identified, but initial exploration into this matter suggests that a specific ZIP code can be attributed to this phenomena.

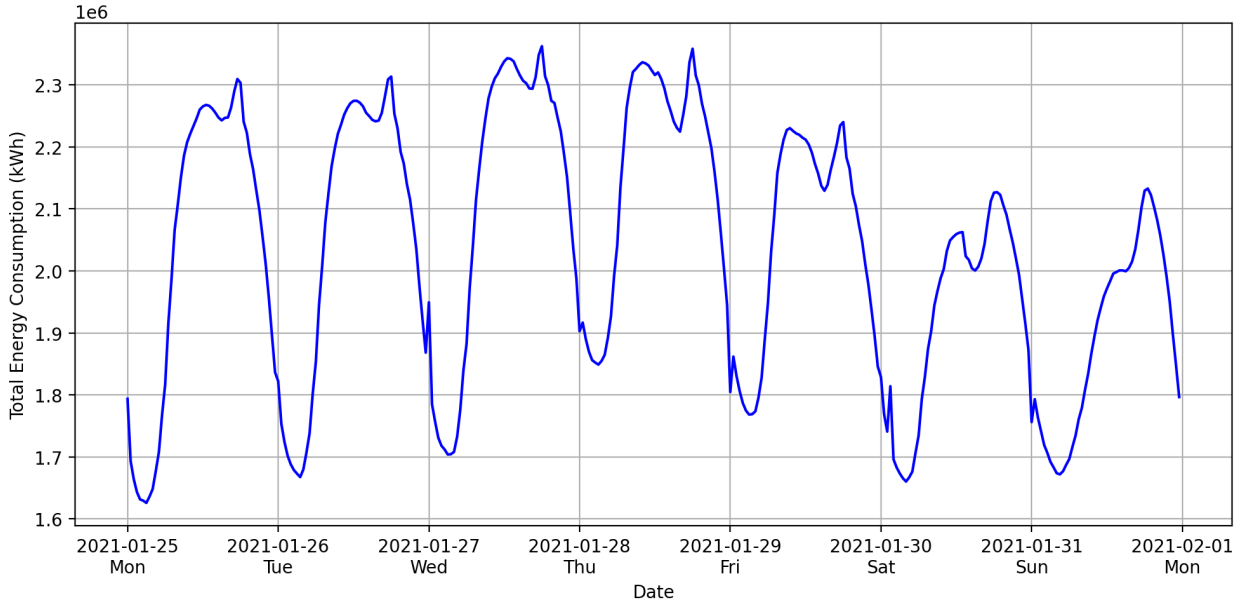

FIGURE 4.4. Semi-Hourly Consumption, January 25$^{\text{th}}$ to January 31$^{\text{st}}$, 2021

4.3 Energy Demand Forecasting

The data presented in Figure 4.4 and the preceding time-series data as a whole are prime candidates for demand forecasting. In fact, energy demand forecasting has been an important topic of research for the past several decades,[6] with substantial efforts made to develop predictive models.

Previous forecasting methods range from regression, autoregressive moving average (ARMA) models and its variants, to artificial neural network models and genetic algorithm approaches. Kim and Cho[7] obtained promising results using a novel Convolutional Neural Network/ Long Short Term Memory deep learning model to predict household energy consumption, and the latter is the technique of choice for the research.

4.4 Long Short Term Memory (LSTM) Models

Long Short Term Memory (LSTM) models are a family of deep learning Recurrent Neural Networks (RNN) used to predict sequential data. LSTMs were developed to address limitations with traditional RNN models, and are renowned for capturing long range temporal dependencies in time-series data.

An LSTM layer is comprised of memory cells, which contains components that dictate the flow of sequential data in the model. These components consist of: an input gate, which accepts new data; a forget gate, that discards unnecessary data; the cell and hidden states,

which store the LSTM's memory; and the output gate, which determines the output of a memory cell.

More concretely, suppose there is an input sequence of data $X = \{x_1, x_2, \ldots, x_t\}$ corresponding to t time steps. For $x_t \in X$ the LSTM layer computes the following functions:

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

Where h_t is the hidden state at time step t, c_t is the cell state at time step t, and h_t is the hidden state at at time step t_1 or the initial hidden state at t_0. Functions i_t, f_t, g_t, o_t are the input, forget, cell, and output gates respectively, and σ is the sigmoid function. The operator $\odot$ is the Hadamard product, the element-wise multiplication of matrices of identical size, where $C_{ij} = A_{ij} \odot B_{ij}$ $\forall i,j$.

4.5 Implementation and Training

Applied to this research, an LSTM RNN model was constructed using the `torch` library to predict energy demand for the data presented in Figure 4.4. The data consists of 336 semi-hourly time steps that were normalized and split to train on the first four days of the data. The model was then tasked to predict the last three days of the week.

The model architecture is comprised of an LSTM layer with 64 features in the hidden state. The LSTM layer was then fed to a complementary dense layer to obtain the predicted forecasting value. Model loss is computed from Mean Square Error (MSE), given by the function:

$$
\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2
$$

and was trained using the ADAM algorithm. The model was trained for 200 epochs and evaluated using RMSE. Upon completion of training, final metrics for the model were a normalized train RMSE of 0.0373 and normalized test RMSE of 0.0385.

Chapter 5 Results and Discussion

5.1 Demand Forecasting Results

The training of the LSTM model constructed in Chapter 4 resulted in the development of an accurate demand forecasting model, able to predict city-wide energy consumption every thirty minutes. Figure 5.1 shows the results of the model, with predicted energy forecasting for the last three days of the week shown in green.

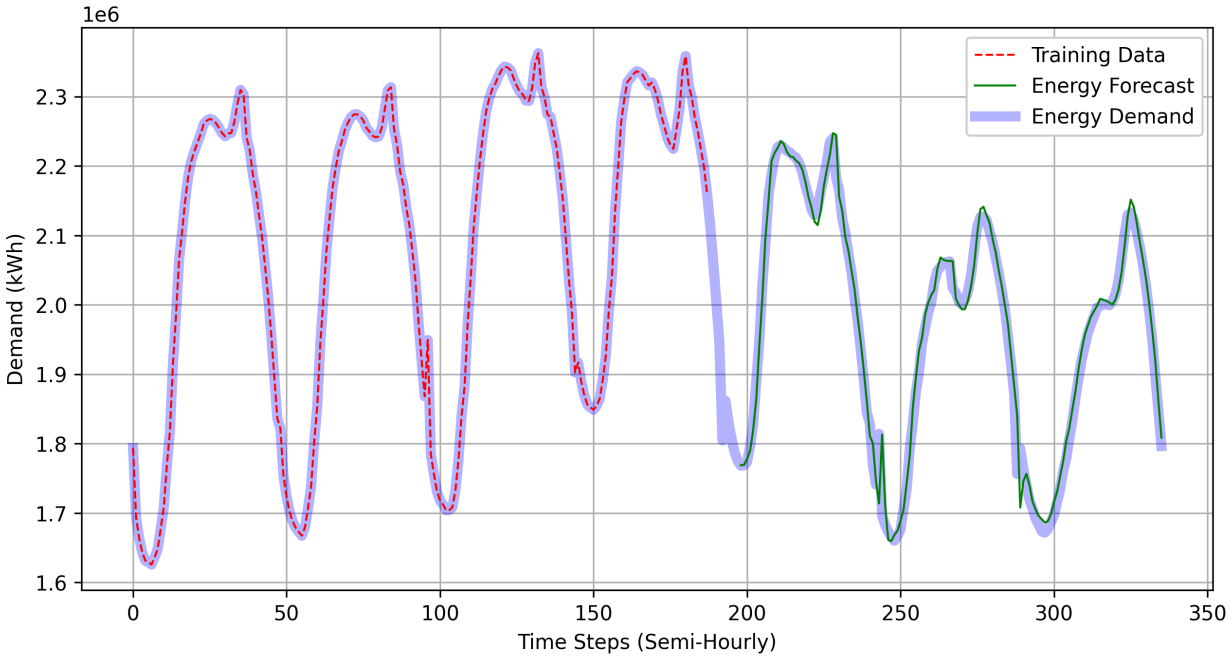

FIGURE 5.1. LSTM Model Forescasting Results

As evident in the figure, the predictive capabilities of the LSTM architecture is apparent. The model is able to capture daily patterns of energy use, conservatively estimating demand for peak AM/PM time periods. More impressively, The model is able to predict the lesser energy demand typical for weekend use patterns, and to replicate the point anomalies present during PM shoulder periods.

5.2 Discussion

The contents of this report shows the promising initial development of deep learning and big data pipeline applied to field of energy infrastructure engineering. With the generation of large volumes of consumption data now standard affair in the energy market, the techniques explored in this research allow for a new method to explore energy consumption trends and patterns. While previous studies have produced similar findings through the use of older frameworks like Spark and Hadoop, the decision to employ **MongoDB** and its associated SaaS services yielded numerous benefits.

In particular, the use of **MongoDB** for data storage, handling, and manipulation resulted in a rapid development and research cycle. The distributed nature and NoSQL architecture of **MongoDB** allowed for dynamic and efficient querying, critical for early exploration of the ComEd data and for structuring research workflow. The accompanying `pymongo` API was intuitive and easy to use, abstracting and simplifying complex operations hosted by Google Cloud Product host services. The aggregation pipeline provided by `pymongo` was similarly easy to use, and was highly configurable for a number of research cases.

Aggregation results generated by **MongoDB** were used to great effect to develop the LSTM energy demand forecasting model. While the results of Chapter 4 were applied to aggregated data for the last week of January 2021 for the entirety of Chicago, Illinois, the data processing pipeline outlined in Chapter 3 can be readily modified to accommodate differing temporal ranges and geospatial regions on-the-fly. While the architecture of the LSTM model was relatively simple, for different analysis needs architecture can be readily configured to match the complexity of varying temporal granularities and spatial resolutions.

While the methodology presented in this research is consistent in approach, there are considerations that must be discussed and acknowledged. As explained in Chapter 2, data preprocessing resulting from the "*15/15*" anonymization protocols precludes the accurate representation of actual energy use and consumption behavior for the project research limits, reducing the scope of this research. Limiting individual customer use to no more than 15% percent of the usage within a data set also prohibits a comprehensive statistical analysis of the data in the context of actual energy use. To extrapolate the results of this research to scale would require the use of auxiliary population data, and an additional layer of geospatial processing to account for spatial mismatch between data sources (population counts for the United States are typically at the spatial resolution of census tract, while the fundamental spatial unit of the research is ZIP code). This factor was not a primary concern given the scope of this research, but should be taken into account in future work.

5.3 Future Work

Despite the limitations of this report, the big data pipeline and machine learning approach developed for this research was constructed with scalability in mind. Leveraging the capabilities of **MongoDB**, the database can be expanded to include the entire 24 months of data provided by ComEd. The data is approximately 500 GB compressed, and the selection of one month of data for analysis was due to time constraints associated with uploading the raw data and financial considerations hosting this data on **MongoDB** (A discount credit code was used to acquire **MongoDB** resources).

To obtain a more realistic representation of total energy use, other smart grid infrastructure components such as phasor measurement units (PMUs), intelligent substation systems, and associated communication infrastructure can provide valuable ancillary data sources to complement smart meter data. The use of NoSQL databases will be paramount in handling these varying data sources, and can prove to be a useful tool to predict and manage future energy demand.

Bibliography

[1] H. Daki, A. El Hannani, A. Aqqal, A. Haidine, and A. Dahbi, "Big Data management in smart grid: concepts, requirements and implementation," *Journal of Big Data*, vol. 4, no. 1, p. 13, Apr. 2017. [Online]. Available: https://doi.org/10.1186/s40537-017-0070-y

[2] J. González Ordiano, A. Bartschat, N. Ludwig, E. Braun, S. Waczowicz, N. Renkamp, N. Peter, C. Düpmeier, R. Mikut, and V. Hagenmeyer, "Concept and benchmark results for Big Data energy forecasting based on Apache Spark," *Journal of Big Data*, vol. 5, no. 1, p. 11, Dec. 2018. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-018-0119-6

[3] S. J. Matthews and A. St. Leger, "Leveraging MapReduce and Synchrophasors for Real-Time Anomaly Detection in the Smart Grid," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 3, pp. 392–403, Jul. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/7902117/

[4] B. Farsi, M. Amayri, N. Bouguila, and U. Eicker, "On Short-Term Load Forecasting Using Machine Learning Techniques and a Novel Parallel Deep LSTM-CNN Approach," *IEEE Access*, vol. 9, pp. 31 191–31 212, 2021, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/document/9356582

[5] J. Rouleau and L. Gosselin, "Impacts of the COVID-19 lockdown on energy consumption in a Canadian social housing building," *Applied Energy*, vol. 287, p. 116565, Apr. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261921001124

[6] I. Ghalehkhondabi, E. Ardjmand, G. R. Weckman, and W. A. Young, "An overview of energy demand forecasting methods published in 2005–2015," *Energy Systems*, vol. 8, no. 2, pp. 411–447, May 2017. [Online]. Available: https://doi.org/10.1007/s12667-016-0203-y

[7] T.-Y. Kim and S.-B. Cho, "Predicting residential energy consumption using CNN-LSTM neural networks," *Energy*, vol. 182, pp. 72–81, Sep. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544219311223